PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 17/30

A1

(11) International Publication Number: WO 99/22315

(43) International Publication Date: 6 May 1999 (06.05.99)

(21) International Application Number: PCT/US98/20719

(22) International Filing Date: 2 October 1998 (02.10.98)

(30) Priority Data:
08/959,058 28 October 1997 (28.10.97) US

(71) Applicant: CACHEFLOW, INC. [US/US]; 650 Almanor Avenue, Sunnyvale, CA 94086 (US).

(72) Inventors: MALCOLM, Michael; 250 Family Farm Road, Woodside, CA 94062 (US). TELFORD, Ian; 72 John Street East, Waterloo, Ontario N2J 1G1 (US).

(74) Agent: SWERNOFSKY, Steven, A.; The Law Offices of Steven A. Swernofsky, P.O. Box 390013, Mountain View, CA 94039–0013 (US).

(81) Designated States: CA, CN, JP, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.

(54) Title: ADAPTIVE ACTIVE CACHE REFRESH

(57) Abstract

The invention provides a system and system for automatically refreshing documents in a cache, so that each particular document is refreshed no more often and no less often than needed. For each document, the cache estimates a probability distribution of times for client requests for that document and a probability distribution of times for server changes to that document. Times for refresh are selected for each particular document in response to both the estimated probability distribution of times for client requests and the estimated probability distribution of times for server changes. The invention also provides a system and system for objectively estimating the value the cache is providing for the system including the cache. The cache estimates for each document a probability distribution of times for client requests for that document, and determines a cumulative probability distribution which reflects the estimated marginal hit rate at the storage limit of the cache and the marginal advantage of adding storage to the cache.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

Title of the Invention

Adaptive Active Cache Refresh

Background of the Invention

1. *Field of the Invention*

This invention relates to caches.

2. *Related Art*

When a client device seeks to obtain information from server devices on a network, it is sometimes desirable to provide a cache, that is, a device which maintains copies of that information so that multiple requests for the same information can be satisfied at the cache, and do not require that information to be transmitted repeatedly across the network. Known caches do this to reduce the amount of communication bandwidth used between the clients and the servers, and when shared by more than one client, act to reduce the total amount t of communication bandwidth used between all of the clients and the servers.

One problem in the art is that information requested a second time (possibly requested a second time by the same client, or requested by a second client after a first client has already requested that information once) can change at the server between the time it is first requested and the time it is requested again. In such cases, transmitting the stored information from the cache gives inaccurate information to the second requester. This can reduce the confidence users at the client devices have for the information provided by the cache.

One known method is to transmit each request from the client device to the server. so as to obtain an answer as to whether the document must be refreshed before it is served (transmitted) to the client device. While this method achieves the purpose of serving only refreshed information to the client device, it has the drawback that the client device must wait for contact with the server device and the reply from the server device, even when the information is already present in the cache. Moreover, this method uses communication bandwidth by sending requests to the server device and receiving confirmations from the server device which can be unnecessary.

It would be advantageous to provide a cache which reduces the average amount of time users at the client device wait for information, rather than attempting to reduce the amount of communication bandwidth between the client devices and the server devices. One aspect of the invention is to automatically refresh the information maintained in the cache, notwithstanding that this uses additional communication bandwidth. The cache occasionally queries the server device to determine if the document has been changed, so the cache can maintain an up-to-date version of the document. When the document is requested by the client device, the cache serves that document immediately without checking with the server device.

Refreshing information in the cache is useful, but some documents require refresh more often than others. If a particular document is selected for refresh less often than required, it will sometimes be served to the client device even though it is "stale" (that is, modified at the server since the cache last obtained a copy). In contrast, if a particular document is selected for refresh more often than required, the document will sometimes be refreshed unnecessarily, thus wasting communication bandwidth.

Accordingly, it would be advantageous to provide a method and system for refreshing documents so that each particular document is refreshed no more often and no less often than needed. This advantage is achieved in a system in which the times for refresh are tailored to each particular document, in response to both an estimated probability distribution of times for client requests for that document and an estimated probability distribution of times for server changes to that document.

Another problem in the art is that it is difficult to objectively determine the value the cache is providing for the system including the cache, or whether the cache is too small or too large. In contrast with persistent storage devices, for which it is easy to determine how full they are and whether the size of the storage device is too small or too large, the cache is nearly always nearly full of data being stored for later request by the client device.

One known method to determine the value of the cache is to measure the cache "hit rate," that is, the fraction of information requests which are for documents already maintained in the cache. However, this measure is extremely dependent on the degree of locality of reference to information requested by the client device, which in turn is extremely dependent on

the number of client devices, the nature of information they are requesting, and the rate at which they request that information.

Accordingly, it would be advantageous to provide a method and system for objectively estimating the value the cache is providing for the system including the cache, such as whether the cache is too small or too large for selected objectives. This advantage is achieved in a system in which the cache estimates for each document a probability distribution of times for client requests for that document, and determines a cumulative probability distribution which reflects the estimated marginal hit rate at the storage limit of the cache and the marginal advantage of adding storage to the cache.

Summary of the Invention

The invention provides a method and system for automatically refreshing documents in a cache, so that each particular document is refreshed no more often and no less often than needed. For each document, the cache estimates a probability distribution of times for client requests for that document and a probability distribution of times for server changes to that document. Times for refresh are selected for each particular document in response to both the estimated probability distribution of times for client requests and the estimated probability distribution of times for server changes.

The invention also provides a method and system for objectively estimating the value the cache is providing for the system including the cache. The cache estimates for each document a probability distribution of times for client requests for that document, and determines a cumulative probability distribution which reflects the estimated marginal hit rate at the storage limit of the cache and the marginal advantage of adding storage to the cache.

Brief Description of the Drawings

Figure 1 shows a block diagram of a system for periodically refreshing documents in a cache.

Figure 2 shows a process flow diagram of a method for periodically refreshing documents in a cache.

Detailed Description of the Preferred Embodiment

In the following description, a preferred embodiment of the invention is described with regard to preferred process steps and data structures. Those skilled in the art would recognize after perusal of this application that embodiments of the invention can be implemented using general purpose processors or special purpose processors or other circuits adapted to particular process steps and data structures described herein, and that implementation of the process steps and data structures described herein would not require undue experimentation or further invention.

Inventions disclosed herein can be used in conjunction with inventions disclosed in one or more of the following patent applications:

Provisional U.S. Application 60/048,986, filed June 9, 1997, in the name of inventors Michael Malcolm and Robert Zarnke, titled "Network Object Cache Engine," assigned to CacheFlow, Inc., attorney docket number CASH-001.

U.S. Application Serial No. 08/______, filed this same day, in the name of inventors Doug Crow, Bert Bonkowski, Harold Czegledi, and Tim Jenks, titled "Shared Cache Parsing and Pre-fetch," assigned to CacheFlow, Inc., attorney docket number CASH-004.

These applications are referred to herein as the "Cache Disclosures," and are hereby incorporated by reference as if fully set forth herein.

System Elements

Figure 1 shows a block diagram of a system for periodically refreshing documents in a cache.

A system 100 includes a cache 110, at least one client device 120, and at least one server device 130. Each client device 120 is coupled to the cache 110 using a client communication path 121, such as a dial-up connection, a LAN (local area network), a WAN (wide area network), or some combination thereof. Similarly, each server device 130 is also coupled to the cache 110 using a server communication path 131, such as a dial-up connection, a LAN (local

area network), a WAN (wide area network), or some combination thereof. In a preferred embodiment, the client communication path 121 includes a LAN, while the server communication path 131 includes a network of networks such as an internet or intranet.

As used herein, the terms "client" and "server" refer to a relationship between the client or server and the cache 110, not necessarily to particular physical devices. As used herein, one "client device" 120 or one "server device" 130 can comprise any of the following: (a) a single physical device capable of executing software which bears a client or server relationship to the cache 110; (b) a portion of a physical device, such as a software process or set of software processes capable of executing on one hardware device, which portion of the physical device bears a client or server relationship to the cache 110; or (c) a plurality of physical device, or portions thereof, capable of cooperating to form a logical entity which bears a client or server relationship to the cache 110. The phrases "client device" 120 and "server device" 130 refer to such logical entities and not necessarily to particular individual physical devices.

The server device 130 includes memory or storage 132 having a web document 133. In a preferred embodiment, the web document 133 can include text and directions for display, pictures, such as data in GIF or JPEG format, other multimedia data, such as animation, audio (such as streaming audio), movies, video (such as streaming video), program fragments, such as Java, Javascript, or ActiveX, or other web documents, such as when using frames.

The cache 110 includes a processor 111, program and data memory 112, and mass storage 113. The cache 110 maintains a first set of web objects 114 in the memory 112 and a second set of web objects 114 in the storage 113.

In a preferred embodiment, the cache 110 includes a cache device such as described in the Cache Disclosures defined herein, hereby incorporated by reference as if fully set forth therein.

The cache 110 receives requests from the client device 120 for a web object 114 and determines if that web object 114 is present at the cache 110, either in the memory 112 or in the storage 113. If the web object 114 is present in the memory 112, the cache 110 transmits the web object 114 to the client device 120 using the client communication path 121. If the web object 114 is present in the storage 113 but not in the memory 112, the cache 110 loads the web object 114 into the memory 112 from the storage 113, and proceeds as in the case when the web

object 114 was originally present in the memory 112. If the web object 114 is not present in either the memory 112 or the storage 113, the cache 110 retrieves the web object 114 from the appropriate server device 130, places the web object 114 in the memory 112 and the storage 113, and proceeds as in the case when the web object 114 was originally present in the memory 112.

Due to the principle of locality of reference, it is expected that the cache 110 will achieve a substantial "hit rate," in which many requests from the client device 120 for web objects 114 will be for those web objects 114 already maintained by the cache 110, reducing the need for requests to the server device 130 using the server communication path 131.

Determining Expected Frequencies

For each web object 114, the cache 110 determines a probability that the web object 114 is stale at time t, and a probability that the web object 114 is requested at request h:

$$\text{Psi (t)} = \text{Probability} \{ \text{ object i is stale at time t } \} \qquad (141)$$

$$\text{Pri (h)} = \text{Probability} \{ \text{ object i is requested at request h } \} \qquad (142)$$

The probability of request Pri (h) is measured for "request h" rather than for "time t," so that the probability Pri (h) is not distorted by changes in the frequency of requests. The frequency of requests can change, for example, while the cache 110 is being moved from one location to another, such as its original shipment, or while there are relatively few client devices 120 requesting web objects 114 from the cache 110, such as early in the morning or late at night. The probability Pri (h) is thus responsive to "request time" **h** rather than "wall-clock time" **t**.

In a preferred embodiment, the cache 110 maintains a value for "request time" **h**, since a request-time of zero when the cache 110 was first manufactured, and stores that value in a non-volatile memory.

Having defined the probabilities Psi (t) and Pri (h), the probability that the web object 114 will be served stale by the cache 110 on the next request is the product of the probabilities Psi (t) • Pri (h).

$$\text{Pi (current time, current request)} = \text{Psi (current time)} \bullet \text{Pri (current request)}$$

= Probability { object i is stale at this time **and**
object i is requested at this request-time }

(143)

Thus, each web object 114 i has a corresponding product Pi (current time, current request-time), which indicates the probability that the web object 114 will be served stale by the cache at the next request. The sum of such products Pi (current time, current request-time) for all web objects 114 i in the cache 110 is the cumulative probability that the next web object 114 requested by one of the client devices 120 will be served stale by the cache 110.

The cache 110 chooses to attempt to refresh the web object 114 with the highest such product Pi (current time, current request-time). The cache 110 automatically attempts to refresh web objects 114 until the cumulative probability of all products Pi (current time, current request-time) is less than a selected threshold value. In a preferred embodiment, the selected threshold value is between about 1% and about 5%.

The probabilities Psi (t) and Pri (h) each follow a Poisson distribution, that is, that the probability of a particular web object 114 becoming stale in any particular time interval is responsive to a random process having a Poisson distribution, the probability of a particular web object 114 being requested in any particular request-time interval is also responsive to a random process having a Poisson distribution.

Accordingly, the cache 110 estimates Psi (t) as follows:

$$\text{Psi}(t) = 1 - \exp(-a\,t) \qquad (144)$$

where the function **exp** is exponentiation using the natural base e, and the value **a** is a parameter of the Poisson process.

It follows that

$$A = \ln(2) / \text{EUI} \qquad (145)$$

where ln (2) is the natural logarithm of 2 (approximately 0.69315), and EUI is the estimated mean interval between updates to the web object 114 i at the

server device 130.

EUI and similar values described herein are specific to each web object 114 i, and are determined separately for each web object 114 i. However, for convenience in notation, the term "EUI" and similar terms are not subscripted to so indicate.

Accordingly, the cache 110 estimates EUI in response to the times of actual updates of the web object 114 at the server device 130, and is able to determine Psi (t) in response to EUI.

Similarly, the cache 110 estimates Pri (h) as follows:

$$\text{Pri}(h) = b \exp(-b\, h) \qquad (146)$$

where
the value **b** is a parameter of the Poisson process.

It follows that

$$\text{Pri}(\text{current request-time}) = \ln(2) / \text{EAI} \qquad (147)$$

where EAI is the estimated mean interval between requests for the web object 114 i at the cache 110 (that is, from any client device 120).

Accordingly, the cache 110 estimates EAI in response to the request-times of actual requests for the web object 114 from any client device 120, and is able to determine Pri (h) in response to EAI.

Method of Operation

Figure 2 shows a process flow diagram of a method for periodically refreshing documents in a cache.

A method 200 includes a set of flow points to be noted. and steps to be executed, cooperatively by the system 100, including the cache 110, the client device 120. and the server device 130.

At a flow point 210, a particular web object 114 is loaded into the cache 110. The particular web object 114 can be loaded into the cache 110 in one of two ways:

Initial load: The web object 114 was first requested from the cache 110 by one of the client devices 120, found not to be maintained in the cache 110, requested by the cache 110 from one of the server devices 130, transmitted by the server device 130 to the cache 110, stored by the cache 110, and transmitted by the cache 110 to the requesting client device 120; or

Reload: The web object 114 was maintained in the cache 110 after a previous initial load or reload, found by the cache 110 to be stale, and reloaded by the cache 110 from one of the server devices 130.

At a step 211, the cache 110 makes an initial estimate of the values of EUI and EAI for the particular web object 114.

In a preferred embodiment, the cache 110 performs the step 211 by determining a type for the web object 114, and making its initial estimate of the values of EUI and EAI in response to that type. The cache 110 determines the type for the web object 114 in response to information in the HTTP (hypertext transfer protocol) response made by the server device 130. The web object 114 can be one of the following types:

type T: a web object 114 for which an expiration time (XT), sometimes called a "time to live" or "TTL," is specified by the server device 130.

type M: a web object 114 for which a last modified time (MT) is specified. but no XT is specified, by the server device 130;

type N: a web object 114 for which no MT or XT is specified by the server device 130.

In a preferred embodiment, the cache 110 estimates a value for EUI in response to the type determined for the web object 114.

For the initial load of the web object 114 into the cache 110, the following values are used:

type T: EUI = max (XT – LT, 900) seconds (151)

where the function **max** indicates the maximum of the values.

type M: EUI = max (10% of (LT –MT), 900) seconds (152)

type N: EUI = 900 seconds

(153)

In a preferred embodiment, the values 10% and 900 are parameters which can be set by an operator for the cache 110. In alternative embodiments, the 10% and 900 can be determined by the cache 110 responsive to global parameters of the cache 110. For example, the value 900 may be replaced by an average of EUI for all web objects 114 maintained in the cache 110.

The values shown herein are just initial estimates for EUI for each web document 114. In alternative embodiments, there might be many other ways to select initial estimates for EUI for each web document 114.

Similarly, in a preferred embodiment, the cache 110 estimates a value for EAI. In a preferred embodiment, the cache 110 makes an initial estimate for EAI equal to the mean value for EAI for all web documents 114 maintained in the cache 110.

At a step212, the cache 110 determines which of the web objects 114 to refresh.

The cache 110 performs the step 212 by determining the probabilities Psi (t) and Pri (h) for each web object 114 i, and selecting for refresh the web object 114 i with the largest product Pi (current time, current request-time).

The cache 110 refreshes the selected web object 114 and repeats the step 212 so long as the cumulative sum of the products Pi (current time, current request-time) is larger than a

selected threshold. As noted herein, in a preferred embodiment the selected threshold is between about 1% and about 5%, although a wide range of other values are likely also to be workable.

At a flow point 220, a selected refresh time (RT) for a particular web object 114 is reached. The selected refresh time equals the latest load time (LT) plus EUI.

At a step 221, the cache 110 updates its estimated EUI for the web object 114, in response to an update history for the web object 114, as follows:

$$\text{new EUI} = (1 - \text{alpha})\,(\text{old EUI}) + (\text{alpha})\,(\text{UT} - \text{LT}) \qquad (171)$$

where UT = time the web object 114 is actually updated at the cache 110, and
LT = time the web object 114 is actually loaded or reloaded at the cache 110.

In a preferred embodiment, the value of the smoothing constant alpha is about 0.4, but a wide range of values between 0 and 1 are likely to be usable.

In a preferred embodiment, EUI is only updated when the web object 114 is actually updated at the cache 110. However, in alternative embodiments, EUI may be updated at other events, such as when the selected refresh time RT exceeds the sum (LT + EUI), that is, the web object 114 is not actually updated until after the estimated mean refresh time RT. More generally, in alternative embodiments, EUI may be updated responsive to (1) the last time at which the web object 114 was actually updated, (2) the amount of time actually passed since that update, and (3) any earlier estimate for EUI.

At a flow point 230, the particular web object 114 is requested by at least one client device 120.

At a step 231, the cache 110 updates its estimated EAI for the web object 114, in response to an request history for the web object 114, as follows:

$$\text{new EAH} = (1 - \text{alpha})\,(\text{old EAH}) + (\text{alpha})\,(\text{AH} - \text{LH}) \qquad (181)$$

where AH = request-time the web object 114 is actually requested by one of the client devices 120, and

LH = request-time the web object 114 was last requested by one of the client devices 120.

In a preferred embodiment, EAI is only updated when the web object 114 is actually requested by one of the client devices 120. However, in alternative embodiments, EAI may be updated at other events, such as when the time between requests exceeds EAI, that is, the web object 114 is not actually requested until after the estimated mean request interval. More generally, in alternative embodiments, EAI may be updated responsive to (1) the last time at which the web object 114 was actually requested, (2) the amount of request-time actually passed since that request, and (3) any earlier estimate for EAI.

In a preferred embodiment, the value of the smoothing constant alpha is about 0.4, but a wide range of values between 0 and 1 are likely usable.

At a flow point 240, the cache 110 is ready to determine an estimated cache hit rate.

At a step 241, the cache 110 sums the values of Pri (current request-time) for all web objects 114 in the cache 110.

The sum of all Pri (current request-time) is an estimate of the probability that the next web object 114 to be requested is one of the web objects 114 maintained in the cache 110, that is, an estimate of the probability of a cache hit.

At a flow point 250, the cache 110 is ready to select a particular web object 114 for removal.

At a step 251, the cache 110 determines which of the web objects 114 to remove.

The cache 110 performs the step 213 by determining the load duration for the web object 114.

$$\text{new LD} = (1 - \text{alpha})\,(\text{old LD}) + (\text{alpha})\,(c + t) \qquad (191)$$

where alpha = a smoothing constant, preferably about 0.25.

c = actual connection time to the server device 130 for this load or reload. and

t = actual transmission time to the server device 130 for this load or reload.

In a preferred embodiment, the value 0.25 for alpha is a parameter which can be set by an operator for the cache 110.

In alternative embodiments, the value LD can be estimated by independently estimating LDc and LDt, as follows:

$$LD = LDc + LDt \qquad (192)$$

where LDc = estimated connection time to the server device 130, and

LDt = estimated transfer time of the web object 114 from the server device 130.

Each time the web object 114 is loaded or reloaded from the server devicfe 130, the cache 110 revises its estimates for LDc and LDt, as follows:

$$\text{new } LDc = (1 - alpha)\ (\text{old } LDc) + (alpha)\ (c) \qquad (193)$$

$$\text{new } LDt = (1 - alpha)\ (\text{old } LDt) + (alpha)\ (t) \qquad (194)$$

The cache 110 uses the estimated load duration LD for the web object 114, the size s of the web object 114, and the probability Pri (h) for each web object 114 i. and selects the web object 114 with the smallest product (LD/s) • Pri (current request) for removal.

At a step 252, the cache 110 adds the value Pri (current request) it determined for the web object 114 to one of a set of summation buckets.

In a preferred embodiment, each summation bucket is used to sum about 10,000 sequential values of Pri (h). In an environment in which the mean size for web documents 114 is about eight kilobytes and each disk drive is about two gigabytes, each bucket therefore represents the marginal value of about 1/25 of a disk drive.

In a preferred embodiment, there are about 50 summation buckets. which are selected and filed in a round-robin manner. In an environment in which the mean size for web

documents 114 is about eight kilobytes and each disk drive is about two gigabytes. the et of 50 buckets therefore represents the marginal value of about two disk drives.

At a step 253, the cache 110 deletes the web object 114 and its estimated values EUI and EAH.

Alternative Embodiments

Although preferred embodiments are disclosed herein, many variations are possible which remain within the concept, scope, and spirit of the invention, and these variations would become clear to those skilled in the art after perusal of this application.

Claims

1. A method of operating a cache, including the step of automatically refreshing a set of objects maintained in said cache in response to an estimate for each particular object in said set that said object is stale.

2. A method as in claim 1, wherein said step of automatically refreshing is responsive to an estimate for said particular object that said object will be requested soon.

3. A method as in claim 1, wherein said step of automatically refreshing includes the steps of

for each said object. estimating a probability that said object will be next requested and will also be stale: and

automatically refreshing a first object in response to said probability that said object will be next requested and will also be stale.

4. A method as in claim 3, wherein said first object has a largest said probability that said object will be next requested and will also be stale.

5. A method as in claim 3, wherein said step of determining includes the steps of

estimating a first probability distribution of client requests for said object;

estimating a second probability distribution of server changes to said object; and

estimating said probability that said object will be next requested and will also be stale in response to said first probability distribution and said second probability distribution.

6. A method as in claim 3, wherein said step of estimating said probability that said object will be next requested and will also be stale is responsive to a product of said first probability distribution and said second probability distribution.

7. A method as in claim 3, wherein said step of estimating includes the step of updating said estimated probability distribution for each object in response to a request history for said object.

8. A method as in claim 7, wherein said step of updating includes the steps of

determining an initial estimated probability distribution of client requests for said object: and

updating said estimated probability distribution of client requests for said object in response to said request history.

9. A method as in claim 7, wherein said step of updating includes the steps of

determining an initial estimated probability distribution of client requests for said object;

determining a new estimated probability distribution of client requests for said object in response to said request history: and

determining a composite value of said initial estimated probability distribution and said new estimated probability distribution.

10. A method as in claim 3, wherein said step of determining includes the step of updating said probability for each object in response to an update history for said object.

11. A method as in claim 10, wherein said step of updating includes the steps of

determining an initial estimated probability distribution of server changes to said object; and

updating said estimated probability distribution of server changes to said object in response to said update history.

12. A method as in claim 10, wherein said step of updating includes the step of determining an initial estimated probability distribution of server changes to said object in response to information provided with said object by a server device.

13. A method as in claim 10, wherein said step of updating includes the steps of

determining an initial estimated probability distribution of server changes to said object:

determining a new estimated probability distribution of server changes to said object in response to said update history; and

determining a composite value of said initial estimated probability distribution and said new estimated probability distribution.

14. A method of operating a cache, including the step of automatically refreshing a set of objects maintained in said cache in response to an estimate for each particular object in said set that said object will be requested soon.

15. A method as in claim 14, wherein said step of automatically refreshing includes the steps of

for each said object, estimating a probability that said object will be next requested and will also be stale; and

automatically refreshing a first object in response to said probability that said object will be next requested and will also be stale.

16. A method of operating a cache, including the steps of

Determining, for each object in a set of objects maintained in said cache, a probability of client requests for said object and an expected load duration for said object; and

Selecting a particular object for removal in response to said probability of client requests for said object and said expected load duration for said object.

17. A method as in claim 16, wherein said step of selecting is responsive to a product of load duration per unit size and said probability of client requests for each said object.

18. A method as in claim 16, wherein said step of determining includes the step of updating said probability for each object in response to a request history for said object.

19. A method as in claim 18, wherein said step of updating includes the steps of

determining an initial estimated probability distribution of client requests for said object; and

updating said estimated probability distribution of client requests for said object in response to said request history.

20. A method as in claim 18, wherein said step of updating includes of steps of

determining an initial estimated probability distribution of client requests for said object;

determining a new estimated probability distribution of client requests for said object in response to said request history; and

determining a composite value of said initial estimated probability distribution and said new estimated probability distribution.

21. A method as in claim 16, wherein said step of determining includes the step of updating said expected load duration for each object in response to a history for said object.

22. A method as in claim 21, wherein said step of updating includes the steps of

determining an initial expected load duration for said object; and

updating said expected load duration for said object in response to said request history.

23. A method as in claim 21, wherein said step of updating includes the steps of

determining an initial expected load duration for said object;

determining a new expected load duration for said object in response to said history; and

determining a composite value of said initial expected load duration and said new expected load duration.

24. A method as in claim 21, wherein said step of updating includes the step of updating said expected load duration for a particular object when said object reloaded.

25. A method of operating a cache, including the steps of

for a plurality of objects maintained in said cache, determining an estimated probability distribution of cliente requests for said object; and

cumulating an element of said estimated probability distribution for a set of said objects.

26. A method as in claim 25, wherein said set of said objects includes a set of recently deleted objects.

27. A system, including

a cache; and

means for automatically refreshing a set of objects maintained in said cache in response to an estimate for each particular object in said set that said object is stale.

28. A system as in claim 27, wherein said means for automatically refreshing is responsive to an estimate for said particular object that said object will be requested soon.

29. A system as in claim 27, wherein said means for automatically refreshing includes

means for estimating a probability, for each said oabject, that said object will be next requested and will also be stale; and

means for automatically refreshing a first object in response to said probability that said object will be next requested and will also be stale.

30. A system as in claim 29, wherein said first object has a largest said probability that said object will be next requested and will also be stale.

31. A system as in claim 29, wherein said means for determining includes

memory storing an estimate of a first probability distribution of client requests for said object;

memory storing an estimate of a second probability distribution of server changes to said object; and

means for estimating said probability that said object will be next requested and will also be stale in response to said first probability distribution and said second probability distribution.

32. A system as in claim 29, wherein said means for estimating said probability that said object will be next requested and will also be stale is responsive to a product of said first probability distribution and said second probability distribution.

33. A system as in claim 29, wherein said means for estimating includes means for updating said estimated probability distribution for each object in response to a request history for said object.

34. A system as in claim 33, wherein said means for updating includes

memory storing an initial estimated probability distributio of client requests for said; and

means for updating said estimated probability distribution of client requests for said object in response to said request history.

35. A system as in claim 33, wherein said means for updating includes means for estimating an initial estimated probability distribution of client requests for said object;

means for estimating a new estimated probability distribution of client requests for said object in response to said request history; and

means for determining a composite value of said initial estimated probability distribution and said new estimated probability distribution.

36. A system as in claim 29, wherein said means for determining includes means for updating said probability for each object in response to an update history for said object.

37. A system as in claim 36, wherein said means for updating includes

Means for estimating an initial estimated probability distribution of server changes to said object; and

means for updating said estimated probability distribution of server changes to said object in response to said update history.

38. A system as in claim 36, wherein said means for updating includes means for estimating an initial estimated probability distribution of server changes to said object in response to information provided with said object by a server device.

39. A system as in claim 36, wherein said means for updating includes

means for estimating an initial estimated probability distribution of server changes to said object;

means for estimating a new estimated probability distribution of server changes to said object in response to said update history; and

means for determining a composite value of said initial estimated probability distribution and said new estimated probability distribution.

40. A system, including

a cache; and

means for automatically refreshing a set of objects maintained in said cache in response to an estimate for each particular object in said set that said object will be requested soon.

41. A system as in claim 40, wherein said means for automatically refreshing includes the steps of

for each said object, estimating a probability that said object will be next requested and will also be stale; and

automatically refreshing a first object in response to said probability that said object will be next requested and will also be stale.

42. A system for operating a cache, including

means for determining, for each object in a set of objects maintained in said cache, a probability of client requests for said object and an expected load duration for said object; and

means for selecting a particular object for removal in response to said probability of client requests for said object and said expected load duration for said object.

43. A system as in claim 42, wherein said means for selecting is responsive to a product of load duration per unit size and said probability of client requests for each said object.

44. A system as in claim 42, wherein said means for determining includes means for updating said probability for each object in response to a request history for said object.

45. A system as in claim 44, wherein said means for updating includes means for estimating an initial estimated probability distribution of client requests for said object; and

means for updating said estimated probability distribution of client requests for said object in response to said request history.

46. A system as in claim 44, wherein said means for updating includes means for estimating an initial estimated probability distribution of client requests for said object;

means for estimating a new estimated probability distribution of client requests for said object in response to said request history; and

means for determining a composite value of said initial estimated probability distribution and said new estimated probability distribution.

47. A system as in claim 42, wherein said means for determining includes means for updating said expected load duration for each object in response to a history for said object.

48. A system as in claim 47, wherein said means for updating includes means for estimating an initial expected load duration for said object; and

means for updating said expected load duration for said object in response to said request history.

49. A system as in claim 47, wherein said means for updating includes means for estimating an initial expected load duration for said object;

means for estimating a new expected load duration for said object in response to said history; and

means for determining a composite value of said initial expected load duration and said new expected load duration.

50. A system as in claim 47, wherein said means for updating includes means for updating said expected load duration for a particular object when said object is reloaded.

51. A system for operating a cache, including means for determining, for each object in a set of objects maintained in said cache, an estimated probability distribution of client requests for said object; and

means for cumulating an element of said estimated probability distribution for a set of said objects.

52. A system as in claim 51, wherein said set of said objects includes a set of recently deleted objects.

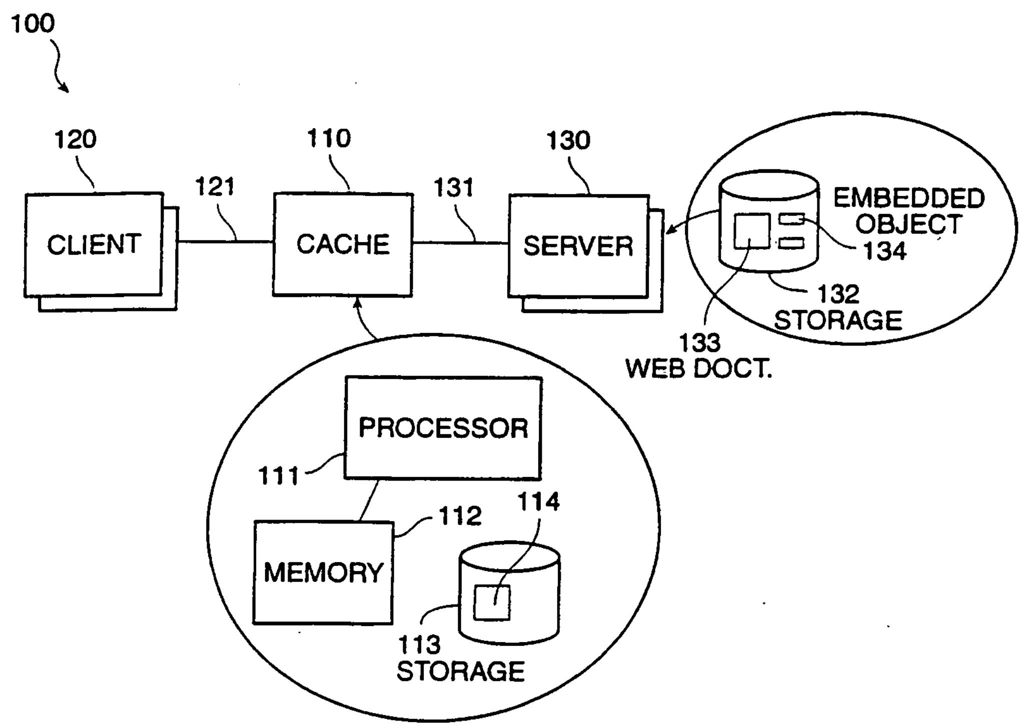
100
120
CLIENT
121
110
CACHE
131
130
SERVER
EMBEDDED OBJECT
134
132
STORAGE
133
WEB DOCT.
PROCESSOR
111
112
114
MEMORY
113
STORAGE

FIG. 1

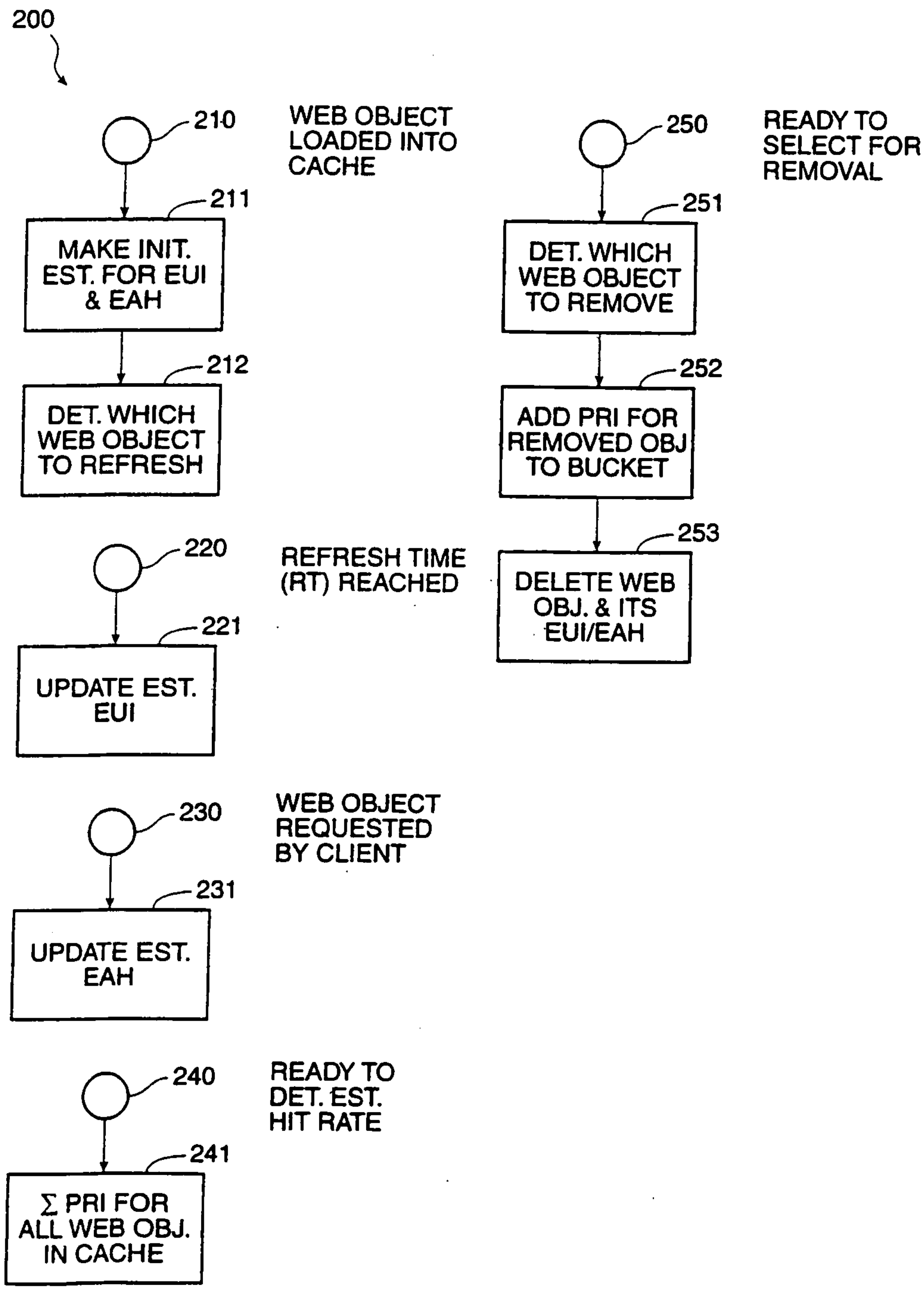
200
210
WEB OBJECT LOADED INTO CACHE
211
MAKE INIT. EST. FOR EUI & EAH
212
DET. WHICH WEB OBJECT TO REFRESH
220
REFRESH TIME (RT) REACHED
221
UPDATE EST. EUI
230
WEB OBJECT REQUESTED BY CLIENT
231
UPDATE EST. EAH
240
READY TO DET. EST. HIT RATE
241
Σ PRI FOR ALL WEB OBJ. IN CACHE
250
READY TO SELECT FOR REMOVAL
251
DET. WHICH WEB OBJECT TO REMOVE
252
ADD PRI FOR REMOVED OBJ TO BUCKET
253
DELETE WEB OBJ. & ITS EUI/EAH

FIG. 2

INTERNATIONAL SEARCH REPORT

Inte Application No
PCT/US 98/20719

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06F17/30

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	DIAS G. ET AL: "A Smart Internet Caching System" PROCEEDINGS OF THE INET'96 CONFERENCE, MONTREAL, CANADA, 24 - 28 June 1996, XP002086721 htt://www.isoc.org/inet96/proceedings/a4/a4_3.htm	1-3,5, 14,15, 27-29, 31,40,41
A	see the whole document	4,6-13, 16-26, 30, 32-39, 42-52

	-/--	

[X] Further documents are listed in the continuation of box C.

[ ] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
4 December 1998	21/12/1998

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Fournier, C

1

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Inter Application No
PCT/US 98/20719

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
X	GLASSMAN S: "A caching relay for the World Wide Web" COMPUTER NETWORKS AND ISDN SYSTEMS, vol. 27, no. 2, November 1994, page 165-173 XP004037987 see page 166, right-hand column, line 23 - page 167, right-hand column, line 3 see page 172, right-hand column, paragraph 11 - page 173, left-hand column, paragraph 12	1,2,14, 27,28,40
A	ZHIMEI JIANG ET AL: "Prefetching links on the WWW" 1997 IEEE INTERNATIONAL CONFERENCE ON COMMUNICATIONS. TOWARDS THE KNOWLEDGE MILLENNIUM. ICC '97. CONFERENCE RECORD (CAT. NO.97CH36067), PROCEEDINGS OF ICC'97 - INTERNATIONAL CONFERENCE ON COMMUNICATIONS, MONTREAL, QUE., CANADA, 8-12 JUNE 1997, pages 483-489 vol.1, XP002086568 ISBN 0-7803-3925-8, 1997, New York, NY, USA, IEEE, USA see page 483, left-hand column, paragraph 1 - page 484, right-hand column, paragraph 3A see page 4898, left-hand column, paragraph 6	1-52
A	DINGLE A ET AL: "Web cache coherence" COMPUTER NETWORKS AND ISDN SYSTEMS, vol. 28, no. 11, May 1996, page 907-920 XP004018195 see page 907, left-hand column, paragraph 1 - page 911, right-hand column, paragraph 2.5 see page 917, left-hand column, paragraph 4 - page 918, left-hand column, paragraph 4.2	1-52
A	NABESHIMA M: "The Japan Cache Project: an experiment on domain cache" COMPUTER NETWORKS AND ISDN SYSTEMS, vol. 29, no. 8-13, September 1997, page 987-995 XP004095297 see page 991, left-hand column, paragraph 4.2 - page 992, right-hand column, paragraph 4.2.5; figure 2	1-52

1